

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants: Hee-Kwan SON Conf: Unknown  
Application No.: NEW Group: Unknown  
Filed: September 12, 2003 Examiner: Unknown  
For: MONTGOMERY MODULAR MULTIPLIER USING A  
COMPRESSOR AND MULTIPLICATION METHOD

**PRIORITY LETTER**

September 12, 2003

Honorable Commissioner of Patents and Trademarks  
Washington, DC 20231

Dear Sirs:

Pursuant to the provisions of 35 U.S.C. 119, enclosed is/are a certified copy of the following priority document(s).

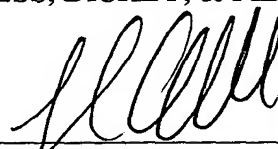
<u>Application No.</u>	<u>Date Filed</u>	<u>Country</u>
10-2002-0087243	December 30, 2002	REPUBLIC OF KOREA

In support of Applicant's priority claim, please enter this document into the file.

Respectfully submitted,

HARNESS, DICKEY, & PIERCE, P.L.C.

By

  
\_\_\_\_\_  
John A. Castellano, Reg. 35,094  
P.O. Box 8910  
Reston, Virginia 20195  
(703) 390-3030

JAC/mh

# 대한민국 특허청

## KOREAN INTELLECTUAL PROPERTY OFFICE

별첨 사본은 아래 출원의 원본과 동일함을 증명함.

This is to certify that the following application annexed hereto  
is a true copy from the records of the Korean Intellectual  
Property Office.

출원 번호 : 10-2002-0087243  
Application Number

출원 년 월 일 : 2002년 12월 30일  
Date of Application DEC 30, 2002

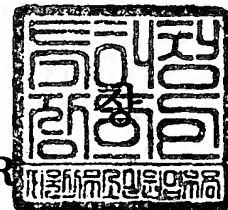
출원인 : 삼성전자주식회사  
Applicant(s) SAMSUNG ELECTRONICS CO., LTD.



2003 년 04 월 07 일

특 허 청

COMMISSIONER



## 【서지사항】

【서류명】	특허출원서
【권리구분】	특허
【수신처】	특허청장
【참조번호】	0027
【제출일자】	2002.12.30
【국제특허분류】	G06F
【발명의 명칭】	4-2 컴프레서를 이용한 몽고메리 모듈러 승산기 및 그 승산 방법
【발명의 영문명칭】	Montgomery modular multiplier by 4 to 2 compressor and multiplication method thereof
【출원인】	
【명칭】	삼성전자 주식회사
【출원인코드】	1-1998-104271-3
【대리인】	
【성명】	이영필
【대리인코드】	9-1998-000334-6
【포괄위임등록번호】	1999-009556-9
【대리인】	
【성명】	정상빈
【대리인코드】	9-1998-000541-1
【포괄위임등록번호】	1999-009617-5
【발명자】	
【성명의 국문표기】	손희관
【성명의 영문표기】	SON, Hee Kwan
【주민등록번호】	690311-1551517
【우편번호】	442-400
【주소】	경기도 수원시 팔달구 망포동 동수원엘지빌리지 106-1101
【국적】	KR
【심사청구】	청구
【취지】	특허법 제42조의 규정에 의한 출원, 특허법 제60조의 규정에 의한 출원심사를 청구합니다. 대리인 이영필 (인) 대리인 정상빈 (인)

**【수수료】**

【기본출원료】 20 면 29,000 원

【가산출원료】 15 면 15,000 원

【우선권주장료】 0 건 0 원

【심사청구료】 15 항 589,000 원

【합계】 633,000 원

**【첨부서류】**

1. 요약서·명세서(도면)\_1통

## 【요약서】

## 【요약】

4-2 컴프레서를 이용한 몽고메리 모듈러 승산기 및 그 승산 방법이 개시된다. 상기 몽고메리 모듈러 승산기는, A, B, 모듈러 계수 M, 캐리 C, 합산 수 S 각각의 비트 값,  $a_i$ ,  $b_i$ ,  $m_i$ ,  $c_i$ , 및  $s_i$ 을 저장하는 레지스터들을 구비하고, "mod M"에 대하여 "ABR<sup>-1</sup>"의 컨그루언트를 계산하는 몽고메리 모듈러 승산기로서, 먼저,  $b_i A$  계산 논리 회로가 상기 A와 상기  $b_i$ 를 승산하여, 비트별로  $b_i A$ 를 출력한다. 이때,  $q_i$  계산 논리 회로는 " $s_0 \text{ XOR } c_0 \text{ XOR } (b_i \text{ AND } a_0)$ "를 연산하여, 그 결과 값  $q_i$ 를 출력하고,  $q_i M$  계산 논리 회로는 상기 M과 상기  $q_i$ 를 승산하여, 비트별로  $q_i M$ 을 출력한다. 이에 따라, 4-2 컴프레서는 캐리 전파 애더 신호(ONCPA)에 따라, 캐리 저장 애더 구조(CSA)에 의하여 상기 S의 중간 계산 값 및 상기 C의 중간 계산 값을 비트별로 계산하고, 캐리 전파 애더 구조(CPA)에 의하여 상기 중간 계산 값들을 합산하여 상기 S의 최종 결과와 상기 C의 최종 결과를 S 및 C 레지스터에 출력한다. 따라서, 연산 속도가 빠르고 파워-딜레이 곱(power-delay product)이 작아 몽고메리 모듈러 승산 알고리즘 연산 성능을 향상시키는 효과가 있다.

## 【대표도】

도 1

**【명세서】****【발명의 명칭】**

4-2 컴프레서를 이용한 몽고메리 모듈러 승산기 및 그 승산 방법{Montgomery modular multiplier by 4 to 2 compressor and multiplication method thereof}

**【도면의 간단한 설명】**

본 발명의 상세한 설명에서 인용되는 도면을 보다 충분히 이해하기 위하여 각 도면의 간단한 설명이 제공된다.

도 1은 본 발명의 일실시예에 따른 몽고메리(Montgomery) 모듈러 승산기(modular multiplier)의 블록도이다.

도 2는 도 1의 4-2 컴프레서(compressor)와 그 주변 회로의 구체적인 블록도이다.

도 3은 본 발명의 일실시예에 따른 몽고메리(Montgomery) 모듈러 승산기(modular multiplier)의 동작 설명을 위한 흐름도이다.

도 4는 본 발명의 일실시예에 따른 몽고메리(Montgomery) 모듈러 승산기(modular multiplier)의 캐리 저장 애더(carry save adder) 동작 설명을 위한 도면이다.

도 5는 본 발명의 일실시예에 따른 몽고메리(Montgomery) 모듈러 승산기(modular multiplier)의 캐리 전파 애더(carry propagation adder) 동작 설명을 위한 도면이다.

**【발명의 상세한 설명】****【발명의 목적】****【발명이 속하는 기술분야 및 그 분야의 종래기술】**

- <7> 본 발명은 공개키(public key) 암호 시스템(crypto system)에 관한 것으로, 특히 몽고메리(Montgomery) 모듈러 승산기(modular multiplier)에 관한 것이다.
- <8> 스마트 카드, IC 카드 등을 통한 통신에 사용되는 암호 시스템(crypto system)은, 비밀키 방식에서 공개키(public key) 방식으로 발전해왔다. 비밀키 방식에서는 통신하고자 하는 두 사용자가 동일한 비밀키를 공유해야 하므로 키 관리가 어렵고 디지털 서명을 구현하기 어려운 반면, 공개키(public key) 방식에서는 비밀키만 각 사용자가 보관하고, 공개된 상대방의 공개키(public key)를 알고있는 사용자는 누구든지 그 상대방과 통신을 할 수 있어서 비밀 통신의 편리함을 제공한다.
- <9> 공개키(public key) 암호 시스템(crypto system)에는 RSA(Ron Rivest, Adi Shamir, and Len Adleman), 디피와 헬만(Diffie-Hellman), DSA(Digital Signature Algorithm), 및 ECC(Elliptic Curve Cryptosystem) 등이 있다. 이와 같은 공개키(public key) 암호 시스템(crypto system)은, 모듈러 멍승(modular exponentiation) 연산을 위한 모듈러 승산(modular multiplication)을 기본 연산으로 하고 있다. 따라서, 공개키(public key) 암호 시스템(crypto system)에는 반드시 모듈러 승산기(modular multiplier)가 채용된다.
- <10> 모듈러 승산(modular multiplication)을 하기에 가장 효과적인 알고리즘은 몽고메리(Montgomery) 모듈러 승산(modular multiplication) 알고리즘으로 알려져 있다. 몽고

메리(Montgomery) 모듈러 승산(modular multiplication) 알고리즘을 슈도 코드(pseudo code)로 표시하면, [알고리즘 1]과 같다.

<11> [알고리즘 1]

<12> Stimulus:

<13>  $A = (a_{n-1} \ a_{n-2} \ \dots \ a_1 \ a_0)_2$ , and  $A < M$

<14>  $B = (b_{n-1} \ b_{n-2} \ \dots \ b_1 \ b_0)_2$ , and  $B < M$

<15>  $M = (m_{n-1} \ m_{n-2} \ \dots \ m_1 \ m_0)_2$ , and  $M$  is odd.

<16> Response:

<17>  $S = (S_n \ S_{n-1} \ S_{n-2} \ \dots \ S_1 \ S_0)_2 \equiv ABR^{-1} \pmod{M}$

<18> Method:

<19>  $S := 0$

<20> for  $i := 0$  to  $n-1$  do

<21>  $q_i := s_0 \text{ XOR } (b_i \text{ AND } a_0)$

<22>  $S := (S + b_i A + q_i M) / 2$

<23> endfor

<24> 즉, [알고리즘 1]에서, "for" 루프 안에서 계산되는 최종  $S(\text{sum})$ ([알고리즘 1]에서 캐리를  $S_n$ 으로 표시함)가, "mod  $M$ "에 대하여 " $ABR^{-1}$ "의 컨그루언트(congruent)를 연산한 결과 값으로 된다. 여기서, "mod  $M$ " 대한 모듈러 곱 역수인 " $R^{-1}$ "은,  $R=2^n$ 일 때, " $(R \cdot R^{-1}) \bmod M$ " 값이 "1"로 되도록 하는 수이다.



- <25> 이와 같은 몽고메리(Montgomery) 모듈러 승산(modular multiplication) 알고리즘에서는, 나눗셈 없이 주어진 수들(A,B,M)을 곱셈만으로 연산하는 알고리즘으로서, 다른 알고리즘에 비해 속도가 빠르므로 모듈러 역승(modular exponentiation) 연산이 필요한 공개키(public key) 암호 시스템(crypto system) 구현 시 가장 널리 이용된다.
- <26> 이와 같은 몽고메리(Montgomery) 모듈러 승산(modular multiplication) 알고리즘을 적용하는 종래의 몽고메리(Montgomery) 모듈러 승산기(modular multiplier)는, 캐리 전파 애더(carry propagation adder)를 기본 누산기(accumulator) 구조로 사용하는 패러렐 승산기(parallel multiplier) 형태이거나 3-2(3 inputs to 2 outputs) 컴프레서(compressor), 즉, 풀 애더(full adder)로 이루어지는 캐리 저장 애더(carry save adder)를 기본 누산기(accumulator) 구조로 사용하는 시리얼 승산기(serial multiplier) 형태 등으로 되어있다.
- <27> 그러나, 각 비트의 합산마다 캐리(carry)를 전달하기 위한 한 클럭씩의 전파 지연 시간을 필요로 하는 캐리 전파 애더(carry propagation adder) 구조는, 캐리(carry)의 전파 지연 시간을 무한정 크게 할 수 없으므로 32-비트 이상의 수들에 대한 승산을 수행하기 어려운 문제가 있다. 즉, 캐리 전파 애더(carry propagation adder) 구조는 캐리 저장 애더(carry save adder) 구조에 비하여 파워-딜레이 곱(power-delay product)이 크고, 32-비트 이상의 큰 수들에 대한 승산을 수행하기 위하여는, 32-비트 수와 32-비트 수를 반복적으로 승산하는 방법으로 수행해야 하는 문제가 있다.
- <28> 한편, 각 비트마다 별도로 필요한 모든 합산을 비트 수만큼의 클럭에 수행하여 전파 지연 시간을 제거한 3-2 컴프레서(compressor), 즉, 풀 애더(full adder)를 사용하는 캐리 저장 애더(carry save adder) 구조의 승산기는, 전파 지연 시간 문제가 없으나,

하드웨어로의 구현이 용이하지 않은 문제가 있다. 즉, [알고리즘 1]과 같은 몽고메리(Montgomery) 모듈러 승산(modular multiplication) 알고리즘에서, 합산 할 워드(word)는 4개(캐리,  $S$ ,  $b_iA$ ,  $q_iM$ )인데, 3-2 컴프레서(compressor)에 입력 가능한 수는 3개뿐이므로, 2개의 주 입력 워드(word)( $b_iA$ ,  $q_iM$ )를 미리 더해두는 과정이 필요하다는 문제가 있다. 또한, 3-2 컴프레서(compressor)에서 합산 될 때에는, 캐리(carry),  $S$ (sum), 및 4개(캐리,  $S$ ,  $b_iA$ ,  $q_iM$ )의 워드(word) 중 선택된 어느 하나의 워드(word)가 입력되어야 하므로, 4개의 워드(word)(캐리,  $S$ ,  $b_iA$ ,  $q_iM$ ) 중 어느 하나의 워드(word)를 선택하기 위한 4:1 멀스(multiplexer)도 필요하다는 문제가 있다.

**【발명이 이루고자 하는 기술적 과제】**

<29> 따라서, 본 발명이 이루고자하는 기술적 과제는, 몽고메리(Montgomery) 모듈러 승산(modular multiplication)에 있어서, 4개의 워드(word)(캐리,  $S$ ,  $b_iA$ ,  $q_iM$ )에 대하여 각 비트마다 별도로 필요한 모든 합산을 먼저 수행하는 캐리 저장 애더(carry save adder) 구조로 전파 지연 시간을 제거하고, 최종 출력에서만 캐리 전파 애더(carry propagation adder) 구조로 합산을 수행하여, 연산 속도가 빠르고 파워-딜레이 곱(power-delay product)이 작은 몽고메리(Montgomery) 모듈러 승산기(modular multiplier)를 제공하는 데 있다.

<30> 본 발명이 이루고자하는 다른 기술적 과제는, 몽고메리(Montgomery) 모듈러 승산(modular multiplication)에 있어서, 4개의 워드(word)(캐리,  $S$ ,  $b_iA$ ,  $q_iM$ )에 대하여 각 비트마다 별도로 필요한 모든 합산을 먼저 수행하는 캐리 저장 애더(carry save adder) 구조로 전파 지연 시간을 제거하고, 최종 출력에서만 캐리 전파 애더(carry propagation adder) 구조로 합산을 수행하여, 연산 속도가 빠르고 파워-딜레이 곱(power-delay

product)이 작은 몽고메리(Montgomery) 모듈러 승산(modular multiplication) 방법을 제공하는 데 있다.

### 【발명의 구성 및 작용】

- <31>        상기의 기술적 과제를 달성하기 위한 본 발명에 따른 몽고메리 모듈러 승산기는, "mod M"(여기서, M은 모듈러 계수)에 대하여  $ABR^{-1}$ (여기서, A 및 B는 입력되는 n 비트 길이의 수,  $R^{-1}$ 은 "mod M"에 대한 R의 모듈러 곱 역수)의 컨그루언트를 계산하는 몽고메리 모듈러 승산기에 있어서, A 레지스터, B 레지스터, M 레지스터,  $b_iA$  계산 논리 회로,  $q_i$  계산 논리 회로,  $q_iM$  계산 논리 회로, 4-2 컴프레서, S 레지스터, 및 C 레지스터를 구비한다.
- <32>        상기 A 레지스터는 상기 M보다 작은 상기 A의 비트 값  $a_i$ (여기서, i는 "0"부터 "n-1"까지)을 저장한다.
- <33>        상기 B 레지스터는 상기 M보다 작은 상기 B의 비트 값  $b_i$ (여기서, i는 "0"부터 "n-1"까지)을 저장한다.
- <34>        상기 M 레지스터는 홀수인 상기 M의 비트 값  $m_i$ (여기서, i는 "0"부터 "n-1"까지)을 저장한다.
- <35>        상기  $b_iA$  계산 논리 회로는 상기 A와 상기  $b_i$ 를 승산하여, 비트별로  $b_iA$ 를 출력한다.
- <36>        상기  $q_i$  계산 논리 회로는 불린 논리 방정식, " $s_0 \text{ XOR } c_0 \text{ XOR } (b_i \text{ AND } a_0)$ "(여기서,  $s_0$ 은 S의 LSB,  $c_0$ 은 C의 LSB,  $b_i$ 는 상기 B의 비트 값,  $a_0$ 은 상기 A의 LSB)"를 연산하고, 그 결과 값  $q_i$ (여기서, i는 "0"부터 "n-1"까지)를 출력한다.

- <37>      상기  $q_iM$  계산 논리 회로는 상기  $M$ 과 상기  $q_i$ 를 승산하여, 비트별로  $q_iM$ 을 출력한다.
- <38>      상기 4-2 컴프레서는 캐리 전파 애더 신호에 따라, 캐리 저장 애더 구조에 의하여 상기  $C$ , 상기  $S$ , 상기  $b_iA$ , 및 상기  $q_iM$ 에 대하여  $n$ 번 합산하는 연산을 먼저 수행하여 상기  $S$ 의 중간 계산 값 및 상기  $C$ 의 중간 계산 값을 비트별로 계산하고, 캐리 전파 애더 구조에 의하여 상기 중간 계산 값들을 합산하여 상기  $S$ 의 최종 결과와 상기  $C$ 의 최종 결과를 출력한다.
- <39>      상기  $S$  레지스터는 상기  $S$ 의 비트 값  $s_i$ (여기서,  $i$ 는 "0"부터 " $n-1$ "까지)을 업데이트 시켜 저장한다.
- <40>      상기  $C$  레지스터는 상기  $C$ 의 비트 값  $c_i$ (여기서,  $i$ 는 "0"부터 " $n-1$ "까지)을 업데이트 시켜 저장한다.
- <41>      여기서, 상기 4-2 컴프레서는, 제 1폴 애더, 믹스들, 및 제 2폴 애더를 구비한다.
- <42>      상기 제 1폴 애더는 상기  $b_iA$  중 대응 비트 값, 상기  $S$ 의  $s_{i+1}$ , 및 상기  $C$ 의  $c_i$ 에 대한 합산을 수행하여,  $cA_i$ , 및  $sA_i$ 를 출력한다.
- <43>      상기 믹스들은 상기 캐리 전파 애더 신호에 대응하여, 상기  $q_iM$  중 대응 비트 값, 상기  $sA_i$  및 상기  $cA_{i-1}$ , 또는, 상기  $S$ 의  $s_{i+1}$ , 상기  $C$ 의  $c_i$  및 상기  $C$ 의  $c_{i-1}$ 을 선택적으로 출력한다.
- <44>      상기 제 2폴 애더는 상기 캐리 전파 애더 신호가 비활성화 상태일 때, 상기  $q_iM$  중 대응 비트 값, 상기  $sA_i$  및 상기  $cA_{i-1}$ 에 대하여  $n$ 번 합산하는 연산을 먼저 수행하여 상기  $S$ 의 중간 계산 비트 값  $s_i$  및 상기  $C$ 의 중간 계산 비트 값  $c_i$ 을 비트별로 계산하고,

상기 캐리 전파 애더 신호가 액티브 상태일 때, 상기 S의  $s_{i+1}$ , 상기 C의  $c_i$  및 상기 C의  $c_{i-1}$ 에 대한 합산을 수행하여, 상기 S의 최종 결과와 상기 C의 최종 결과를 출력한다.

<45> 상기 캐리 저장 애더 구조는, 상기 캐리 전파 애더 신호가 비활성화 상태일 때, 상기 제 1폴 애더와 상기 제 2폴 애더가 동시에 동작하여 4 입력 - 2 출력 구조로 동작하는 것을 특징으로 한다.

<46> 상기 캐리 전파 애더 구조는, 상기 캐리 전파 애더 신호가 액티브 상태일 때, 상기 제 2폴 애더만 동작하여 3 입력 - 2 출력 구조로 동작하는 것을 특징으로 한다.

<47> 상기  $cA_{i-1}$ 과 상기  $c_{i-1}$ 은, 그 LSB 값들이 제 1논리 상태인 것을 특징으로 한다.

<48> 상기  $s_{i+1}$ 은, 그 MSB 값이, 상기 캐리 전파 애더 신호가 액티브 상태로 되기 전 클럭에서의 상기  $cA_{n-1}$ 과 같은 것을 특징으로 한다.

<49> 상기의 다른 기술적 과제를 달성하기 위한 본 발명에 따른 몽고메리 모듈러 승산 방법은, A, B, 모듈러 계수 M, 캐리 C, 합산 수 S 각각의 비트 값,  $a_i$ ,  $b_i$ ,  $m_i$ ,  $c_i$ , 및  $s_i$ (여기서,  $i$ 는 "0"부터 " $n-1$ "까지)을 저장하는 레지스터들을 구비하고, "mod M"에 대하여 " $ABR^{-1}$ ( 여기서, A 및 B는 입력되는  $n$  비트 길이의 수,  $R^{-1}$ 은 "mod M"에 대한 R의 모듈러 곱 역수)"의 컨그루언트를 계산하는 몽고메리 모듈러 승산기의 몽고메리 모듈러 승산 방법에 있어서, 다음과 같은 단계를 구비한다.

<50> 즉, 본 발명에 따른 몽고메리 모듈러 승산 방법은, 상기 몽고메리 모듈러 승산기에 의하여 상기 A, 상기 B, 및 상기 M을 수신하는 단계; 상기 몽고메리 모듈러 승산기에 의하여 상기 A와 상기  $b_i$ 를 승산하여, 비트별로  $b_i A$ 를 출력하는 단계; 상기 몽고메리 모듈러 승산기에 의하여 불린 논리 방정식, " $s_0 \text{ XOR } c_0 \text{ XOR } (b_i \text{ AND } a_0)$ "(여기서,  $s_0$ 은 S의

LSB,  $c_0$ 는 C의 LSB,  $b_i$ 는 상기 B의 비트 값,  $a_0$ 은 상기 A의 LSB)"를 연산하고, 그 결과 값  $q_i$ (여기서,  $i$ 는 "0"부터 " $n-1$ "까지)를 출력하는 단계; 상기 몽고메리 모듈러 승산기에 의하여 상기 M과 상기  $q_i$ 를 승산하여, 비트별로  $q_iM$ 을 출력하는 단계; 상기 몽고메리 모듈러 승산기에 의하여 캐리 전파 애더 신호에 따라, 캐리 저장 애더 구조에 의하여 상기 C, 상기 S, 상기  $b_iA$ , 및 상기  $q_iM$ 에 대하여  $n$ 번 합산하는 연산을 먼저 수행하여 상기 S의 중간 계산 값 및 상기 C의 중간 계산 값을 비트별로 계산하는 단계; 및 상기 몽고메리 모듈러 승산기에 의하여 캐리 전파 애더 신호에 따라, 캐리 전파 애더 구조에 의하여 상기 중간 계산 값들을 합산하여 상기 S의 최종 결과와 상기 C의 최종 결과를 출력하는 단계를 구비하는 것을 특징으로 한다.

<51> 여기서, 상기 A는, 상기 M 보다 작은 것을 특징으로 한다.

<52> 상기 B는, 상기 M 보다 작은 것을 특징으로 한다.

<53> 상기 M은, 홀수인 것을 특징으로 한다.

<54> 상기 S의 중간 계산 값, 상기 C의 중간 계산 값, 상기 S의 최종 결과 및 상기 C의 최종 결과는, 상기  $b_iA$  중 대응 비트 값, 상기 S의  $s_{i+1}$ , 및 상기 C의  $c_i$ 에 대한 합산을 수행하여,  $cA_i$ , 및  $sA_i$ 를 출력하는 단계; 상기 캐리 전파 애더 신호에 대응하여, 상기  $q_iM$  중 대응 비트 값, 상기  $sA_i$  및 상기  $cA_{i-1}$ , 또는, 상기 S의  $s_{i+1}$ , 상기 C의  $c_i$  및 상기 C의  $c_{i-1}$ 을 선택적으로 출력하는 단계; 상기 캐리 전파 애더 신호가 비활성화 상태일 때, 상기  $q_iM$  중 대응 비트 값, 상기  $sA_i$  및 상기  $cA_{i-1}$ 에 대하여  $n$ 번 합산하는 연산을 먼저 수행하여 상기 S의 중간 계산 비트 값  $s_i$  및 상기 C의 중간 계산 비트 값  $c_i$ 을 비트별로 계산하는 단계; 및 상기 캐리 전파 애더 신호가 액티브 상태일 때, 상기 S의  $s$

- $i+1$ , 상기 C의  $c_i$  및 상기 C의  $c_{i-1}$ 에 대한 합산을 수행하여 상기 S의 최종 결과와 상기 C의 최종 결과를 출력하는 단계를 구비하여 계산되는 것을 특징으로 한다.
- <55>      상기 캐리 저장 애더 구조는, 상기 캐리 전파 애더 신호가 비활성화 상태일 때, 4 입력 - 2 출력 구조로 합산되어, 상기  $b_iA$  및  $q_iM$ 으로부터, 상기 S의 중간 계산 값, 및 상기 C의 중간 계산 값을 계산하는 것을 특징으로 한다.
- <56>      상기 캐리 전파 애더 구조는, 상기 캐리 전파 애더 신호가 액티브 상태일 때, 3 입력 - 2 출력 구조로 합산되어, 상기 S의 중간 계산 값, 및 상기 C의 중간 계산 값으로부터 상기 S의 최종 결과 및 상기 C의 최종 결과를 계산하는 것을 특징으로 한다.
- <57>      상기  $cA_{i-1}$ 과 상기  $c_{i-1}$ 은, 그 LSB 값들이 제 1논리 상태인 것을 특징으로 한다.
- <58>      상기  $s_{i+1}$ 은, 그 MSB 값이, 상기 캐리 전파 애더 신호가 액티브 상태로 되기 전 클럭에서의 상기  $cA_{n-1}$ 과 같은 것을 특징으로 한다.
- <59>      본 발명과 본 발명의 동작상의 이점 및 본 발명의 실시에 의하여 달성되는 목적을 충분히 이해하기 위해서는 본 발명의 바람직한 실시예를 예시하는 첨부 도면 및 첨부 도면에 기재된 내용을 참조하여야만 한다.
- <60>      이하, 첨부한 도면을 참조하여 본 발명의 바람직한 실시예를 설명함으로써, 본 발명을 상세히 설명한다. 각 도면에 제시된 동일한 참조부호는 동일한 부재를 나타낸다.
- <61>      도 1은 본 발명의 일실시예에 따른 몽고메리(Montgomery) 모듈러 승산기(modular multiplier)의 블록도이다.
- <62>      도 1을 참조하면, 본 발명의 일실시예에 따른 몽고메리(Montgomery) 모듈러 승산기(modular multiplier)는, "mod M"(여기서, M은 모듈러 계수)에 대하여 " $ABR^{-1}$ "(여기

서, A 및 B는 입력되는  $n$  비트 길이의 수,  $R^{-1}$ 은 " $\text{mod } M$ "에 대한  $R$ 의 모듈러 곱 역수"의  
 켄그루언트를 계산하는 몽고메리(Montgomery) 모듈러 승산기(modular multiplier)에 있  
 어서, A 레지스터(110), B 레지스터(120), M 레지스터(130),  $b_i A$  계산 논리 회로(140),  
 $q_i$  계산 논리 회로(150),  $q_i M$  계산 논리 회로(160), 4-2 컴프레서(compressor)(170), S  
 레지스터(180), 및 C 레지스터(190)를 구비한다.

<63>        상기 A 레지스터(110)는 상기 M보다 작은 상기 A의 비트 값  $a_i$ (여기서,  $i$ 는 "0"부  
 터 " $n-1$ "까지)을 저장한다. 즉, A는 입력되는  $n$  비트 길이의 수를 나타내는 하나의  
 워드(word)를 나타내고,  $a_i$ 는 상기 A를 구성하는  $a_0$ 에서  $a_{n-1}$ 까지의 각 비트 값이다.

<64>        상기 B 레지스터(120)는 상기 M보다 작은 상기 B의 비트 값  $b_i$ (여기서,  $i$ 는 "0"부  
 터 " $n-1$ "까지)을 저장한다. 즉, B는 입력되는  $n$  비트 길이의 수를 나타내는 하나의  
 워드(word)를 나타내고,  $b_i$ 는 상기 B를 구성하는  $b_0$ 에서  $b_{n-1}$ 까지의 각 비트 값이다.

<65>        상기 M 레지스터(130)는 홀수인 상기 M의 비트 값  $m_i$ (여기서,  $i$ 는 "0"부터 " $n-1$ "까  
 지)을 저장한다. 즉, 모듈러 계수(modulus) M은 입력되는  $n$  비트 길이의 수를 나타내는  
 하나의 워드(word)를 나타내고,  $m_i$ 는 상기 M을 구성하는  $m_0$ 에서  $m_{n-1}$ 까지의 각 비트 값  
 이다.

<66>        상기  $b_i A$  계산 논리 회로(140)는 상기 A와 상기  $b_i$ 를 승산하여, 비트별로  $b_i A$ 를 출  
 령한다. 여기서, 비트별로  $b_i A$ 를 출력하는 것은,  $b_i a_0$ 에서  $b_i a_{n-1}$ 까지  $n$  비트의 값을 출  
 령하는 것이다. 이때,  $i$ 는 [알고리즘 1] 내의 "for" 루프를 수행하기 위하여 0에서  $n-1$   
 까지 변하므로, 도 1에 도시된 바와 같이, "for" 루프 내의 알고리즘을 한번 수행할 때



마다 오른쪽으로 한 비트씩 쉬프트(right shift) 하는 상기 B 레지스터(120)의 LSB(least significant bit)에서  $b_i$  값이 출력된다.

<67>      상기  $q_i$  계산 논리 회로(150)는, [알고리즘 1]의 "for" 루프 내에 있는  $q_i$  식을 계산하기 위하여, 불린(boolean) 논리 방정식, " $s_0 \text{ XOR } c_0 \text{ XOR } (b_i \text{ AND } a_0)$ "(여기서,  $s_0$ 은 S의 LSB,  $c_0$ 은 C의 LSB,  $b_i$ 는 상기 B의 비트 값,  $a_0$ 은 상기 A의 LSB)"를 연산하고, 그 결과 값  $q_i$ (여기서,  $i$ 는 "0"부터 " $n-1$ "까지)를 출력한다. 이때,  $i$ 는 [알고리즘 1] 내의 "for" 루프를 수행하기 위하여 0에서  $n-1$ 까지 변하므로, 도 1에 도시된 바와 같이, "for" 루프 내의 알고리즘을 한번 수행할 때마다 오른쪽으로 한 비트씩 쉬프트(right shift) 하는 상기 B 레지스터(120)의 LSB(least significant bit)에서  $b_i$  값이 출력된다. 이외에,  $s_0$ ,  $c_0$ , 및  $a_0$  각각은 합산 값 S(sum), 캐리 C, 및 상기 A의 LSB 값이다.

<68>      상기  $q_iM$  계산 논리 회로(160)는 상기 M과 상기  $q_i$ 를 승산하여, 비트별로  $q_iM$ 을 출력한다. 여기서, 비트별로  $q_iM$ 을 출력하는 것은,  $q_i m_0$ 에서  $q_i m_{n-1}$ 까지  $n$  비트의 값을 출력하는 것이다. 이때,  $i$ 는 [알고리즘 1] 내의 "for" 루프를 수행하기 위하여 0에서  $n-1$ 까지 변하므로, 도 1에 도시된 바와 같이, "for" 루프 내의 알고리즘을 한번 수행할 때마다  $i$ 는 1씩 증가하여,  $q_i$ 는  $q_0$ 에서  $q_{n-1}$ 까지 출력된다.

<69>      상기 4-2 컴프레서(compressor)(170)는 캐리 전파 애더 신호(ONCPA)에 따라, 캐리 저장 애더 구조에 의하여 상기 C, 상기 S, 상기  $b_iA$ , 및 상기  $q_iM$ 에 대하여  $n$ 번 합산하는 연산을 먼저 수행하여 상기 S의 중간 계산 값 및 상기 C의 중간 계산 값을 비트별로 계산하고, 캐리 전파 애더 구조에 의하여 상기 중간 계산 값들을 합산하여 상기 S의 최종 결과와 상기 C의 최종 결과를 출력한다. 여기서, 상기 캐리 저장 애더 구조는, 상기

캐리 전파 애더 신호(ONCPA)가 비활성화 상태, 즉, 제 1논리 상태("0")일 때, 상기 제 1 폴 애더와 상기 제 2폴 애더가 동시에 동작하여 4 입력 - 2 출력 구조로 동작하는 것이다. 또한, 상기 캐리 전파 애더 구조는, 상기 캐리 전파 애더 신호(ONCPA)가 액티브 상태, 즉, 제 2논리 상태("1")일 때, 상기 제 2폴 애더만 동작하여 3 입력 - 2 출력 구조로 동작하는 것이다.

<70>        상기 S 레지스터(180)는 상기 S의 비트 값  $s_i$ (여기서,  $i$ 는 "0"부터 " $n-1$ "까지)을 업데이트 시켜 저장한다. 즉, S는 합산되어 출력되는  $n$  비트 길이의 수를 나타내는 하나의 워드(word)를 나타내고,  $s_i$ 는 상기 S를 구성하는  $s_0$ 에서  $s_{n-1}$ 까지의 각 비트 값이다. 여기서, 상기 S는, 상기 4-2 컴프레서(compressor)(170)에서 캐리 저장 애더 구조, 또는 캐리 전파 애더 구조에 의하여 합산이 수행될 때마다 새로운 값으로 업데이트 된다.

<71>        상기 C 레지스터(190)는 상기 C의 비트 값  $c_i$ (여기서,  $i$ 는 "0"부터 " $n-1$ "까지)을 업데이트 시켜 저장한다. 즉, C는 캐리로 출력되는  $n$  비트 길이의 수를 나타내는 하나의 워드(word)를 나타내고,  $c_i$ 는 상기 C를 구성하는  $c_0$ 에서  $c_{n-1}$ 까지의 각 비트 값이다. 여기서, 상기 C는, 상기 4-2 컴프레서(compressor)(170)에서 캐리 저장 애더 구조, 또는 캐리 전파 애더 구조에 의하여 합산이 수행될 때마다 새로운 값으로 업데이트 된다.

<72>        도 2는 도 1의 4-2 컴프레서(compressor)(170)와 그 주변 회로의 구체적인 블록도이다.

<73>        도 2를 참조하면, 상기 4-2 컴프레서(compressor)(170)는, 4입력-2출력을 나타내는 애더(adder)로서, 제 1폴 애더(171), 믹스들(173), 및 제 2폴 애더(175)를 구비한다.

- <74>      상기 제 1폴 애더(171)는 상기  $b_iA$  중 대응 비트 값, 상기 S의  $s_{i+1}$ , 및 상기 C의  $c_i$ 에 대한 합산을 수행하여, 캐리  $cA_i$ , 및 합산 값  $sA_i$ 를 출력한다. 상기  $b_iA$  중 대응 비트 값은  $b_ia_i$ 를 말한다.
- <75>      상기 믹스들(173)은 상기 캐리 전파 애더 신호(ONCPA)에 대응하여, 상기  $q_iM$  중 대응 비트 값, 상기  $sA_i$  및 상기  $cA_{i-1}$ , 또는, 상기 S의  $s_{i+1}$ , 상기 C의  $c_i$  및 상기 C의  $c_{i-1}$ 을 선택적으로 출력한다. 상기  $q_iM$  중 대응 비트 값은  $q_{im\ i}$ 를 말한다.
- <76>      상기 제 2폴 애더(175)는 상기 캐리 전파 애더 신호(ONCPA)가 비활성화 상태, 즉, 제 1논리 상태("0")일 때, 상기  $q_iM$  중 대응 비트 값( $q_{im\ i}$ ), 상기  $sA_i$  및 상기  $cA_{i-1}$ 에 대하여 n번 합산하는 연산을 먼저 수행하여 상기 S의 중간 계산 비트 값  $s_i$  및 상기 C의 중간 계산 비트 값  $c_i$ 를 비트별로 계산하고, 상기 캐리 전파 애더 신호(ONCPA)가 액티브 상태, 즉, 제 2논리 상태("1")일 때, 상기 S의  $s_{i+1}$ , 상기 C의  $c_i$  및 상기 C의  $c_{i-1}$ 에 대한 합산을 수행하여, 상기 S의 최종 결과와 상기 C의 최종 결과를 출력한다.
- <77>      여기서, 상기  $cA_{i-1}$ 은 상기 제 1폴 애더(171)에서 출력되는 하나 하위 비트의 캐리이고, 상기 S의  $s_{i+1}$ 은 상기 제 2폴 애더(175)에서 출력되는 하나 상위 비트의 합산 값이며, 상기 C의  $c_{i-1}$ 은 상기 제 2폴 애더(175)에서 출력되는 하나 하위 비트의 캐리이다.
- <78>      상기  $cA_{i-1}$ 과 상기  $c_{i-1}$ 은, 도 2에 도시된 바와 같이, 그 LSB 값들이 제 1논리 상태이고, 상기  $s_{i+1}$ 은, 그 MSB(maximum significant bit) 값이, 상기 캐리 전파 애더 신호(ONCPA)가 액티브 상태로 되기 전 클럭에서의 상기  $cA_{n-1}$ 과 같다. 상기  $cA_{n-1}$ 은 상기 제 1폴 애더(171)에서 출력되는 상기  $cA_i$  중 MSB 값이다.

- <79> 상기한 바와 같은, 본 발명의 일실시예에 따른 몽고메리(Montgomery) 모듈러 승산기(modular multiplier)의 동작을 좀더 상세하게 설명한다.
- <80> 도 3은 본 발명의 일실시예에 따른 몽고메리(Montgomery) 모듈러 승산기(modular multiplier)의 동작 설명을 위한 흐름도이다.
- <81> 도 3을 참조하면, 본 발명의 일실시예에 따른 몽고메리(Montgomery) 모듈러 승산기(modular multiplier)는, A, B, 모듈러 계수 M, 캐리 C, 합산 수 S 각각의 비트 값,  $a_i$ ,  $b_i$ ,  $m_i$ ,  $c_i$ , 및  $s_i$ (여기서,  $i$ 는 "0"부터 " $n-1$ "까지)을 저장하는 레지스터들을 구비하고, "mod M"에 대하여 " $ABR^{-1}$ (여기서, A 및 B는 입력되는  $n$  비트 길이의 수,  $R^{-1}$ 은 "mod M"에 대한 R의 모듈러 곱 역수)"의 컨그루언트를 계산하는 몽고메리(Montgomery) 모듈러 승산기(modular multiplier)로서, 그 방법은 다음과 같다.
- <82> 즉, 본 발명의 일실시예에 따른 몽고메리(Montgomery) 모듈러 승산기(modular multiplier)는, 먼저, 입력되는 상기 A, 상기 B, 및 상기 M을 수신하고(S311), 변수  $i$ 와 합산 값 S가 저장되는 레지스터는 "0"으로 리셋하여 초기화시킨다(S313). 여기서, 상기 A 및 B는, 상기 M 보다 작고, 상기 M은 홀수이다.
- <83> 다음에, 본 발명의 일실시예에 따른 몽고메리(Montgomery) 모듈러 승산기(modular multiplier)는, [알고리즘 1]의 "for" 루프 내에 있는  $q_i$  식을 계산하기 위하여,  $q_i$  계산 논리 회로(150)에서, 불린 논리 방정식, " $s_0 \text{ XOR } c_0 \text{ XOR } (b_i \text{ AND } a_0)$ "(여기서,  $s_0$ 은 S의 LSB,  $c_0$ 은 C의 LSB,  $b_i$ 는 상기 B의 비트 값,  $a_0$ 은 상기 A의 LSB)"를 연산하고, 그 결과 값  $q_i$ (여기서,  $i$ 는 "0"부터 " $n-1$ "까지)를 출력한다(S315-S319). 또한,  $b_i A$  계산 논리 회로(140)에서는 상기 A와 상기  $b_i$ 를 승산하여, 비트별로  $b_i A$ 를 출력하며,  $q_i M$  계산 논리 회로(160)M 계산

논리 회로(160)에서는 상기 M과 상기  $q_i$ 를 승산하여, 비트별로  $q_iM$ 을 출력한다 (S315~S319). 이와 같이 하여 계산된  $b_iA$ 와  $q_iM$ 으로부터, 4-2 컴프레서(compressor)(170)에서는 캐리 전파 애더 신호(ONCPA)의 비활성화 상태, 즉, 제 1논리 상태("0")에 따라, 캐리 저장 애더 구조에 의하여 상기 C, 상기 S, 상기  $b_iA$ , 및 상기  $q_iM$ 에 대하여 n번 합산하는 연산을 먼저 수행하여 상기 S의 중간 계산 값 및 상기 C의 중간 계산 값을 비트별로 계산한다(S315~S319).

<84> 도 4는 본 발명의 일실시예에 따른 몽고메리(Montgomery) 모듈러 승산기(modular multiplier)의 캐리 저장 애더(carry save adder) 동작 설명을 위한 도면이다.

<85> 도 4를 참조하면,  $b_iA$ 와  $q_iM$ 으로부터, 상기 S의 중간 계산 값, 및 상기 C의 중간 계산 값 계산은, 상기 제 1풀 애더(171)에 의하여, 상기  $b_iA$  중 대응 비트 값, 상기 S의  $s_{i+1}$ , 및 상기 C의  $c_i$ 에 대한 합산을 수행하여,  $cA_i$ , 및  $sA_i$ 를 출력하면, 상기 제 2풀 애더(175)에 의하여, 믹스들(173)에 의하여 선택된 상기  $q_iM$  중 대응 비트 값, 상기  $sA_i$  및 상기  $cA_{i-1}$ 에 대하여 n번 합산하는 연산을 먼저 수행하여 상기 S의 중간 계산 비트 값  $s_i$  및 상기 C의 중간 계산 비트 값  $c_i$ 를 비트별로 계산함으로써 이루어진다 (S315~S319).

<86> 이때, 도 2에 도시된 바와 같이, 상기 C는 각 비트의 출력이 같은 비트의 상기 제 1풀 애더(171)에 입력되고, 상기 S는 각 비트의 출력이 하나 하위 비트의 상기 제 1풀 애더(171)에 입력되도록 하여 [알고리즘 1]의 "for" 루프 내의 2로 나누는 결과 값이 계산된다(S315). 또한, 상기  $cA_{i-1}$ 의 LSB 값은 제 1논리 상태("0")에

있다. 즉, 상기 캐리 저장 애더 구조는, 상기 캐리 전파 애더 신호(ONCPA)가 비활성화 상태일 때, 4 입력 - 2 출력 구조로 합산되어, 상기  $b_iA$  및  $q_iM$ 으로부터, 상기 S의 중간 계산 값, 및 상기 C의 중간 계산 값을 계산하는 것이다.

<87> 다음에, 상기 S의 중간 계산 값 및 상기 C의 중간 계산 값이 비트별로 계산되면, 4-2 컴프레서(compressor)(170)에서는 캐리 전파 애더 신호(ONCPA)의 액티브 상태, 즉, 제 2논리 상태("1")에 따라(S321), 캐리 전파 애더 구조에 의하여 상기 중간 계산 값들을 합산하여 상기 S의 최종 결과와 상기 C의 최종 결과를 출력한다(S323-S327).

<88> 도 5는 본 발명의 일실시예에 따른 몽고메리(Montgomery) 모듈러 승산기(modular multiplier)의 캐리 전파 애더(carry propagation adder) 동작 설명을 위한 도면이다.

<89> 도 5를 참조하면, 상기 중간 계산 값들로부터 상기 S의 최종 결과와 상기 C의 최종 결과의 계산은, 상기 제 2폴 애더(175)에 의하여, 믹스들(173)에 의하여 선택된 상기 S의  $s_{i+1}$ , 상기 C의  $c_i$  및 상기 C의  $c_{i-1}$ 에 대한 합산을 수행하여 상기 S의 최종 결과와 상기 C의 최종 결과를 계산함으로써 이루어진다(S323). 여기서, 상기  $s_{i+1}$ 은, 도 2에 도시된 바와 같이, 그 MSB 값이 상기 캐리 전파 애더 신호(ONCPA)가 액티브 상태로 되기 전 클럭에서의 상기  $cA_{n-1}$ 과 같고, 상기  $c_{i-1}$ 은, 그 LSB 값이 제 1논리 상태("0")에 있다

<90> 이때, 캐리 전파 애더 구조에 의하여 n 비트 전체에 대한 상기 S의  $s_{i+1}$ , 상기 C의  $c_i$  및 상기 C의  $c_{i-1}$ 에 대한 합산을 수행하는 시간은, (한 비트 전파 지연 시간)\*n과 같고, 한 비트 전파 지연 시간은 상기 제 2폴 애더(175)의 전파 지연 시간과 상기 2:1 믹스들(173) 각각의 지연 시간으로 이루어진다(S325). 즉, 상기 캐리 전파 애더 구조는, 상기 캐리 전파 애더 신호(ONCPA)가 액티브 상태일 때, 3 입력 - 2 출력 구조로 합산되

어, 상기 S의 중간 계산 값, 및 상기 C의 중간 계산 값으로부터 상기 S의 최종 결과 및 상기 C의 최종 결과를 계산하는 것이다.

<91> 위에서 기술한 바와 같이, 본 발명의 일실시예에 따른 몽고메리(Montgomery) 모듈러 승산기(modular multiplier)는, A, B, 모듈러 계수 M, 캐리 C, 합산 수 S 각각의 비트 값,  $a_i$ ,  $b_i$ ,  $m_i$ ,  $c_i$ , 및  $s_i$ (여기서,  $i$ 는 "0"부터 " $n-1$ "까지)을 저장하는 레지스터들을 구비하고, "mod M"에 대하여 " $ABR^{-1}$ (여기서, A 및 B는 입력되는  $n$  비트 길이의 수,  $R^{-1}$ 은 "mod M"에 대한 R의 모듈러 곱 역수)"의 컨그루언트를 계산하는 몽고메리(Montgomery) 모듈러 승산기(modular multiplier)로서, 먼저,  $b_iA$  계산 논리 회로(140)가 상기 A와 상기  $b_i$ 를 승산하여, 비트별로  $b_iA$ 를 출력한다. 이때,  $q_i$  계산 논리 회로(150)는 불린 논리 방정식, " $s_0 \text{ XOR } c_0 \text{ XOR } (b_i \text{ AND } a_0)$ "(여기서,  $s_0$ 은 S의 LSB,  $c_0$ 은 C의 LSB,  $b_i$ 는 상기 B의 비트 값,  $a_0$ 은 상기 A의 LSB)"를 연산하여, 그 결과 값  $q_i$ (여기서,  $i$ 는 "0"부터 " $n-1$ "까지)를 출력하고,  $q_iM$  계산 논리 회로(160)M 계산 논리 회로(160)는 상기 M과 상기  $q_i$ 를 승산하여, 비트별로  $q_iM$ 을 출력한다. 이에 따라, 4-2 컴프레서(compressor)(170)는 캐리 전파 애더 신호(ONCPA)에 따라, 캐리 저장 애더 구조에 의하여 상기 C, 상기 S, 상기  $b_iA$ , 및 상기  $q_iM$ 에 대하여  $n$ 번 합산하는 연산을 먼저 수행하여 상기 S의 중간 계산 값 및 상기 C의 중간 계산 값을 비트별로 계산하고, 캐리 전파 애더 구조에 의하여 상기 중간 계산 값들을 합산하여 상기 S의 최종 결과와 상기 C의 최종 결과를 S 및 C 레지스터(190)에 출력한다.

<92> 이상에서와 같이 도면과 명세서에서 최적 실시예가 개시되었다. 여기서 특정한 용어들이 사용되었으나, 이는 단지 본 발명을 설명하기 위한 목적에서 사용된 것이지 의미 한정이나 특허청구범위에 기재된 본 발명의 범위를 제한하기 위하여 사용된 것은

아니다. 그러므로 본 기술 분야의 통상의 지식을 가진 자라면 이로부터 다양한 변형 및 균등한 타 실시예가 가능하다는 점을 이해할 것이다. 따라서, 본 발명의 진정한 기술적 보호 범위는 첨부된 특허청구범위의 기술적 사상에 의해 정해져야 할 것이다.

#### 【발명의 효과】

<93> 상술한 바와 같이 본 발명의 일실시예에 따른 몽고메리(Montgomery) 모듈러 승산기(modular multiplier)는, 몽고메리(Montgomery) 모듈러 승산에 있어서, 4개의 워드(word)(C, S,  $b_iA$ ,  $q_iM$ )에 대하여 각 비트마다 별도로 필요한 모든 합산을 먼저 수행하는 캐리 저장 애더(CSA) 구조로 전파 지연 시간을 제거하고, 최종 출력에서만 캐리 전파 애더(CPA) 구조로 합산을 수행하므로, 연산 속도가 빠르고 파워-딜레이 곱(power-delay product)이 작아 몽고메리(Montgomery) 모듈러 승산 알고리즘 연산 성능을 향상시키는 효과가 있다.



## 【특허청구범위】

## 【청구항 1】

"mod M"(여기서, M은 모듈러 계수)에 대하여 " $ABR^{-1}$ "(여기서, A 및 B는 입력되는 n 비트 길이의 수,  $R^{-1}$ 은 "mod M"에 대한 R의 모듈러 곱 역수)"의 컨그루언트를 계산하는 몽고메리 모듈러 승산기에 있어서,

상기 M보다 작은 상기 A의 비트 값  $a_i$ (여기서, i는 "0"부터 "n-1"까지)을 저장하는 A 레지스터;

상기 M보다 작은 상기 B의 비트 값  $b_i$ (여기서, i는 "0"부터 "n-1"까지)을 저장하는 B 레지스터;

홀수인 상기 M의 비트 값  $m_i$ (여기서, i는 "0"부터 "n-1"까지)을 저장하는 M 레지스터;

상기 A와 상기  $b_i$ 를 승산하여, 비트별로  $b_iA$ 를 출력하는  $b_iA$  계산 논리 회로;

불린 논리 방정식, " $s_0 \text{ XOR } c_0 \text{ XOR } (b_i \text{ AND } a_0)$ "(여기서,  $s_0$ 은 S의 LSB,  $c_0$ 는 C의 LSB,  $b_i$ 는 상기 B의 비트 값,  $a_0$ 은 상기 A의 LSB)"를 연산하고, 그 결과 값  $q_i$ (여기서, i는 "0"부터 "n-1"까지)를 출력하는  $q_i$  계산 논리 회로;

상기 M과 상기  $q_i$ 를 승산하여, 비트별로  $q_iM$ 을 출력하는  $q_iM$  계산 논리 회로;

캐리 전파 애더 신호에 따라, 캐리 저장 애더 구조에 의하여 상기 C, 상기 S, 상기  $b_iA$ , 및 상기  $q_iM$ 에 대하여 n번 합산하는 연산을 먼저 수행하여 상기 S의 중간 계산 값 및 상기 C의 중간 계산 값을 비트별로 계산하고, 캐리 전파 애더 구조에 의하여 상기

중간 계산 값들을 합산하여 상기 S의 최종 결과와 상기 C의 최종 결과를 출력하는 4-2 컴프레서;

상기 S의 비트 값  $s_i$ (여기서,  $i$ 는 "0"부터 " $n-1$ "까지)을 업데이트 시켜 저장하는 S 레지스터; 및

상기 C의 비트 값  $c_i$ (여기서,  $i$ 는 "0"부터 " $n-1$ "까지)을 업데이트 시켜 저장하는 C 레지스터를 구비하는 것을 특징으로 하는 몽고메리 모듈러 승산기.

#### 【청구항 2】

제 1항에 있어서, 상기 4-2 컴프레서는,

상기  $b_iA$  중 대응 비트 값, 상기 S의  $s_{i+1}$ , 및 상기 C의  $c_i$ 에 대한 합산을 수행하여,  $cA_i$ , 및  $sA_i$ 를 출력하는 제 1폴 애더;

상기 캐리 전파 애더 신호에 대응하여, 상기  $q_iM$  중 대응 비트 값, 상기  $sA_i$  및 상기  $cA_{i-1}$ , 또는, 상기 S의  $s_{i+1}$ , 상기 C의  $c_i$  및 상기 C의  $c_{i-1}$ 을 선택적으로 출력하는 믹스들; 및

상기 캐리 전파 애더 신호가 비활성화 상태일 때, 상기  $q_iM$  중 대응 비트 값, 상기  $sA_i$  및 상기  $cA_{i-1}$ 에 대하여  $n$ 번 합산하는 연산을 먼저 수행하여 상기 S의 중간 계산 비트 값  $s_i$  및 상기 C의 중간 계산 비트 값  $c_i$ 을 비트별로 계산하고, 상기 캐리 전파 애더 신호가 액티브 상태일 때, 상기 S의  $s_{i+1}$ , 상기 C의  $c_i$  및 상기 C의  $c_{i-1}$ 에 대한 합산을 수행하여, 상기 S의 최종 결과와 상기 C의 최종 결과를 출력하는 제 2폴 애더를 구비하는 것을 특징으로 하는 몽고메리 모듈러 승산기.

## 【청구항 3】

제 1항 또는 제 2항에 있어서, 상기 캐리 저장 애더 구조는,

상기 캐리 전파 애더 신호가 비활성화 상태일 때, 상기 제 1폴 애더와 상기 제 2폴 애더가 동시에 동작하여 4 입력 - 2 출력 구조로 동작하는 것을 특징으로 하는 몽고메리 모듈러 승산기.

## 【청구항 4】

제 1항 또는 제 2항에 있어서, 상기 캐리 전파 애더 구조는,

상기 캐리 전파 애더 신호가 액티브 상태일 때, 상기 제 2폴 애더만 동작하여 3 입력 - 2 출력 구조로 동작하는 것을 특징으로 하는 몽고메리 모듈러 승산기.

## 【청구항 5】

제 2항에 있어서, 상기  $cA_{i-1}$ 과 상기  $c_{i-1}$ 은,

그 LSB 값들이 제 1논리 상태인 것을 특징으로 하는 몽고메리 모듈러 승산기.

## 【청구항 6】

제 2항에 있어서, 상기  $s_{i+1}$ 은,

그 MSB 값이, 상기 캐리 전파 애더 신호가 액티브 상태로 되기 전 클럭에서의 상기  $cA_{n-1}$ 과 같은 것을 특징으로 하는 몽고메리 모듈러 승산기.

## 【청구항 7】

A, B, 모듈러 계수 M, 캐리 C, 합산 수 S 각각의 비트 값,  $a_i$ ,  $b_i$ ,  $m_i$ ,  $c_i$ , 및  $s_i$  (여기서,  $i$ 는 "0"부터 " $n-1$ "까지)을 저장하는 레지스터들을 구비하고, "mod M"에 대하여 " $ABR^{-1}$ " (여기서, A 및 B는 입력되는  $n$  비트 길이의 수,  $R^{-1}$ 은

"mod M"에 대한 R의 모듈러 곱 역수)의 컨그루언트를 계산하는 몽고메리 모듈러 승산기의 몽고메리 모듈러 승산 방법에 있어서,

상기 몽고메리 모듈러 승산기에 의하여 상기 A, 상기 B, 및 상기 M을 수신하는 단계;

상기 몽고메리 모듈러 승산기에 의하여 상기 A와 상기  $b_i$ 를 승산하여, 비트별로  $b_i A$ 를 출력하는 단계;

상기 몽고메리 모듈러 승산기에 의하여 불린 논리 방정식, " $s_0 \text{ XOR } c_0 \text{ XOR } (b_i \text{ AND } a_0)$ "(여기서,  $s_0$ 은 S의 LSB,  $c_0$ 은 C의 LSB,  $b_i$ 는 상기 B의 비트 값,  $a_0$ 은 상기 A의 LSB)"를 연산하고, 그 결과 값  $q_i$ (여기서,  $i$ 는 "0"부터 " $n-1$ "까지)를 출력하는 단계;

상기 몽고메리 모듈러 승산기에 의하여 상기 M과 상기  $q_i$ 를 승산하여, 비트별로  $q_i M$ 을 출력하는 단계;

상기 몽고메리 모듈러 승산기에 의하여 캐리 전파 애더 신호에 따라, 캐리 저장 애더 구조에 의하여 상기 C, 상기 S, 상기  $b_i A$ , 및 상기  $q_i M$ 에 대하여  $n$ 번 합산하는 연산을 먼저 수행하여 상기 S의 중간 계산 값 및 상기 C의 중간 계산 값을 비트별로 계산하는 단계; 및

상기 몽고메리 모듈러 승산기에 의하여 캐리 전파 애더 신호에 따라, 캐리 전파 애더 구조에 의하여 상기 중간 계산 값들을 합산하여 상기 S의 최종 결과와 상기 C의 최종 결과를 출력하는 단계를 구비하는 것을 특징으로 하는 몽고메리 모듈러 승산 방법.

**【청구항 8】**

제 7항에 있어서, 상기 A는,

상기 M 보다 작은 것을 특징으로 하는 몽고메리 모듈러 승산 방법.

**【청구항 9】**

제 7항에 있어서, 상기 B는,

상기 M 보다 작은 것을 특징으로 하는 몽고메리 모듈러 승산 방법.

**【청구항 10】**

제 7항에 있어서, 상기 M은,

홀수인 것을 특징으로 하는 몽고메리 모듈러 승산 방법.

**【청구항 11】**

제 7항에 있어서, 상기 S의 중간 계산 값, 상기 C의 중간 계산 값, 상기 S의 최종 결과 및 상기 C의 최종 결과는,

상기  $b_i A$  중 대응 비트 값, 상기 S의  $s_{i+1}$ , 및 상기 C의  $c_i$ 에 대한 합산을 수행하여,  $cA_i$ , 및  $sA_i$ 를 출력하는 단계;

상기 캐리 전파 애더 신호에 대응하여, 상기  $q_i M$  중 대응 비트 값, 상기  $sA_i$  및 상기  $cA_{i-1}$ , 또는, 상기 S의  $s_{i+1}$ , 상기 C의  $c_i$  및 상기 C의  $c_{i-1}$ 을 선택적으로 출력하는 단계;

상기 캐리 전파 애더 신호가 비활성화 상태일 때, 상기  $q_iM$  중 대응 비트 값, 상기  $sA_i$  및 상기  $cA_{i-1}$ 에 대하여  $n$ 번 합산하는 연산을 먼저 수행하여 상기  $S$ 의 중간 계산 비트 값  $s_i$  및 상기  $C$ 의 중간 계산 비트 값  $c_i$ 를 비트별로 계산하는 단계; 및

상기 캐리 전파 애더 신호가 액티브 상태일 때, 상기  $S$ 의  $s_{i+1}$ , 상기  $C$ 의  $c_i$  및 상기  $C$ 의  $c_{i-1}$ 에 대한 합산을 수행하여 상기  $S$ 의 최종 결과와 상기  $C$ 의 최종 결과를 출력하는 단계를 구비하여 계산되는 것을 특징으로 하는 몽고메리 모듈러 승산 방법.

#### 【청구항 12】

제 7항 또는 제 11항에 있어서, 상기 캐리 저장 애더 구조는,

상기 캐리 전파 애더 신호가 비활성화 상태일 때, 4 입력 - 2 출력 구조로 합산되어, 상기  $b_iA$  및  $q_iM$ 으로부터, 상기  $S$ 의 중간 계산 값, 및 상기  $C$ 의 중간 계산 값을 계산하는 것을 특징으로 하는 몽고메리 모듈러 승산 방법.

#### 【청구항 13】

제 7항 또는 제 11항에 있어서, 상기 캐리 전파 애더 구조는,

상기 캐리 전파 애더 신호가 액티브 상태일 때, 3 입력 - 2 출력 구조로 합산되어, 상기  $S$ 의 중간 계산 값, 및 상기  $C$ 의 중간 계산 값으로부터 상기  $S$ 의 최종 결과 및 상기  $C$ 의 최종 결과를 계산하는 것을 특징으로 하는 몽고메리 모듈러 승산 방법.

#### 【청구항 14】

제 11항에 있어서, 상기  $cA_{i-1}$ 과 상기  $c_{i-1}$ 은,

그 LSB 값들이 제 1논리 상태인 것을 특징으로 하는 몽고메리 모듈러 승산 방법.

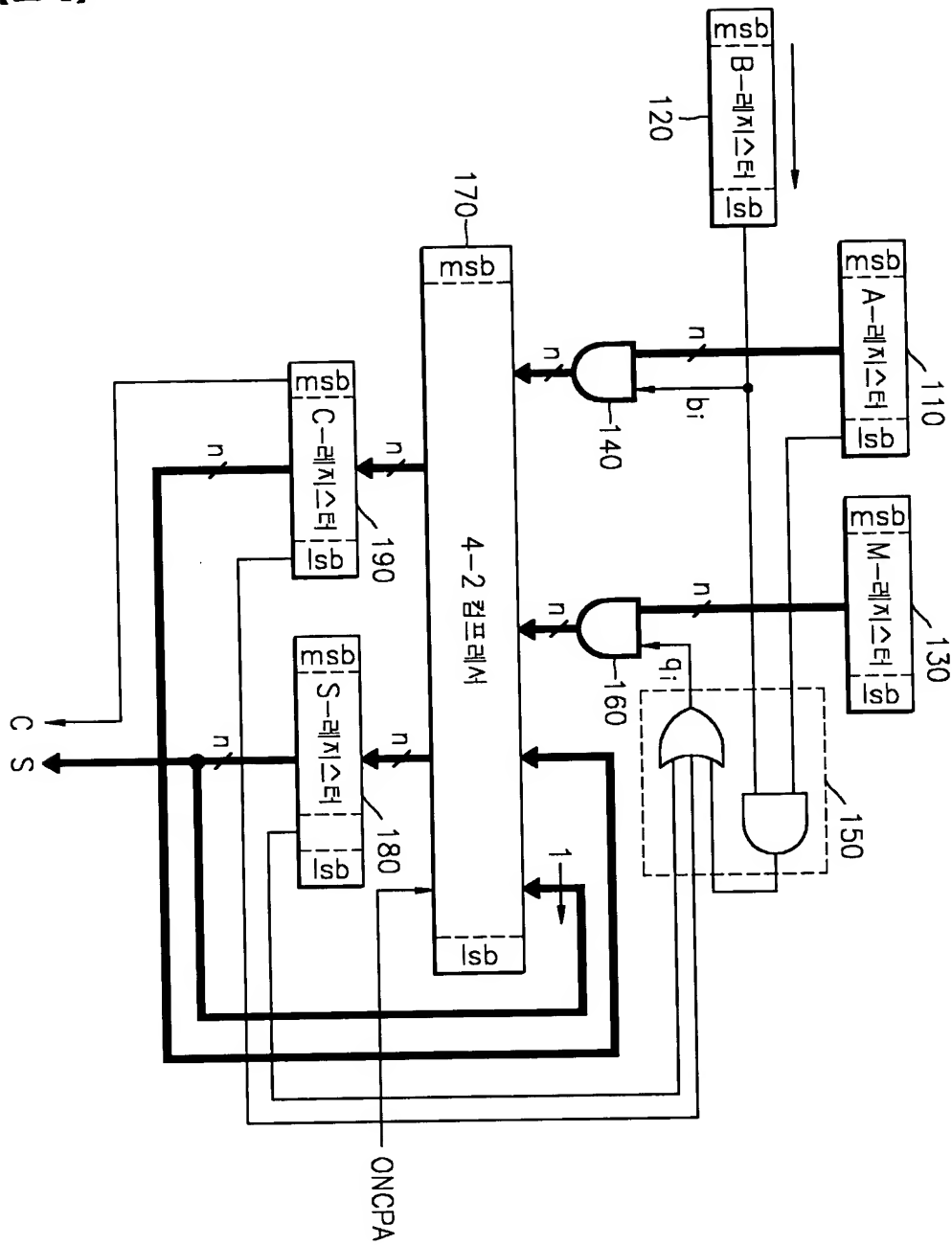
## 【청구항 15】

제 11항에 있어서, 상기  $s_{i+1}$ 은,

그 MSB 값이, 상기 캐리 전파 애더 신호가 액티브 상태로 되기 전 클럭에서의 상기  $cA_{n-1}$ 과 같은 것을 특징으로 하는 몽고메리 모듈러 승산 방법.

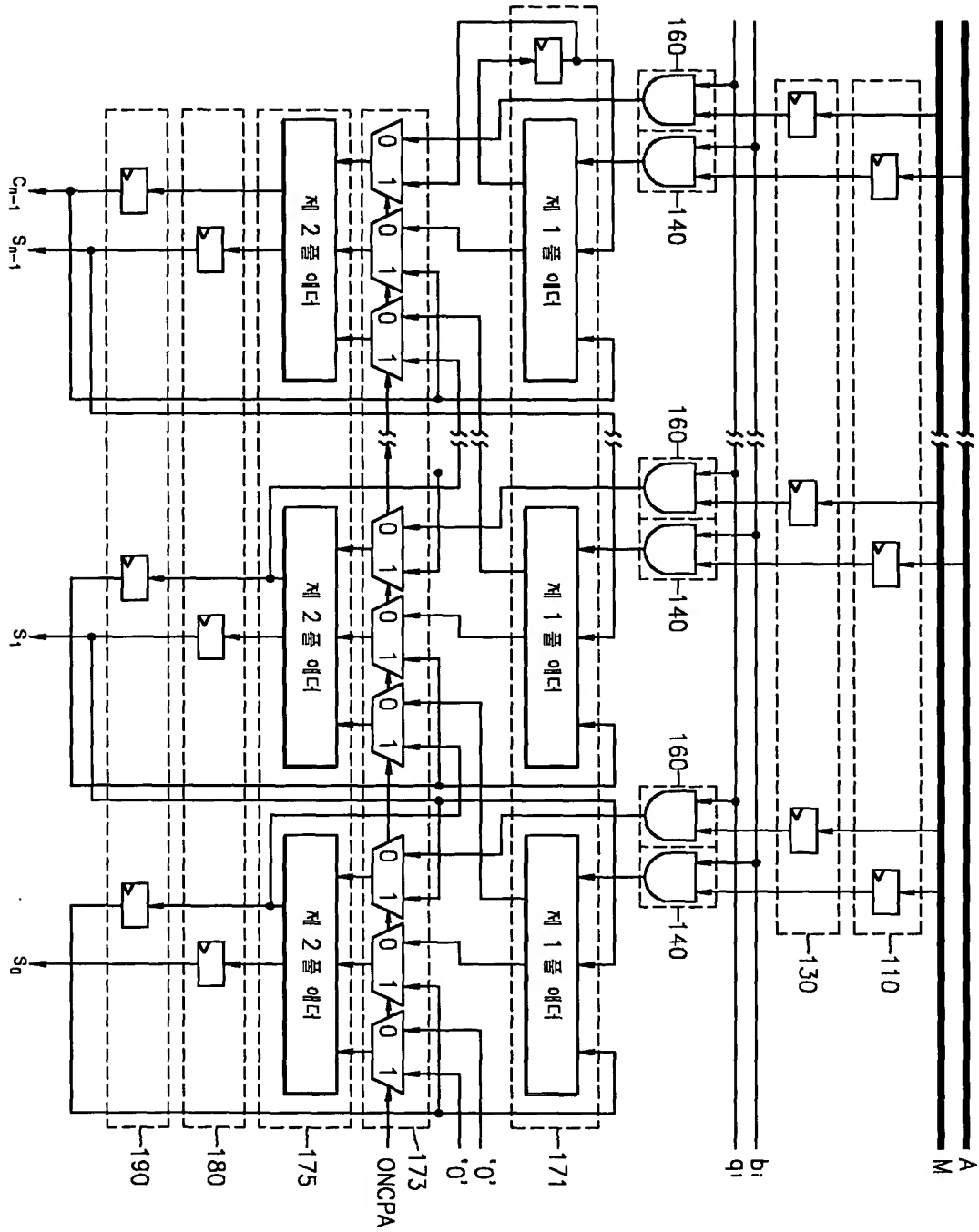
【도면】

【도 1】

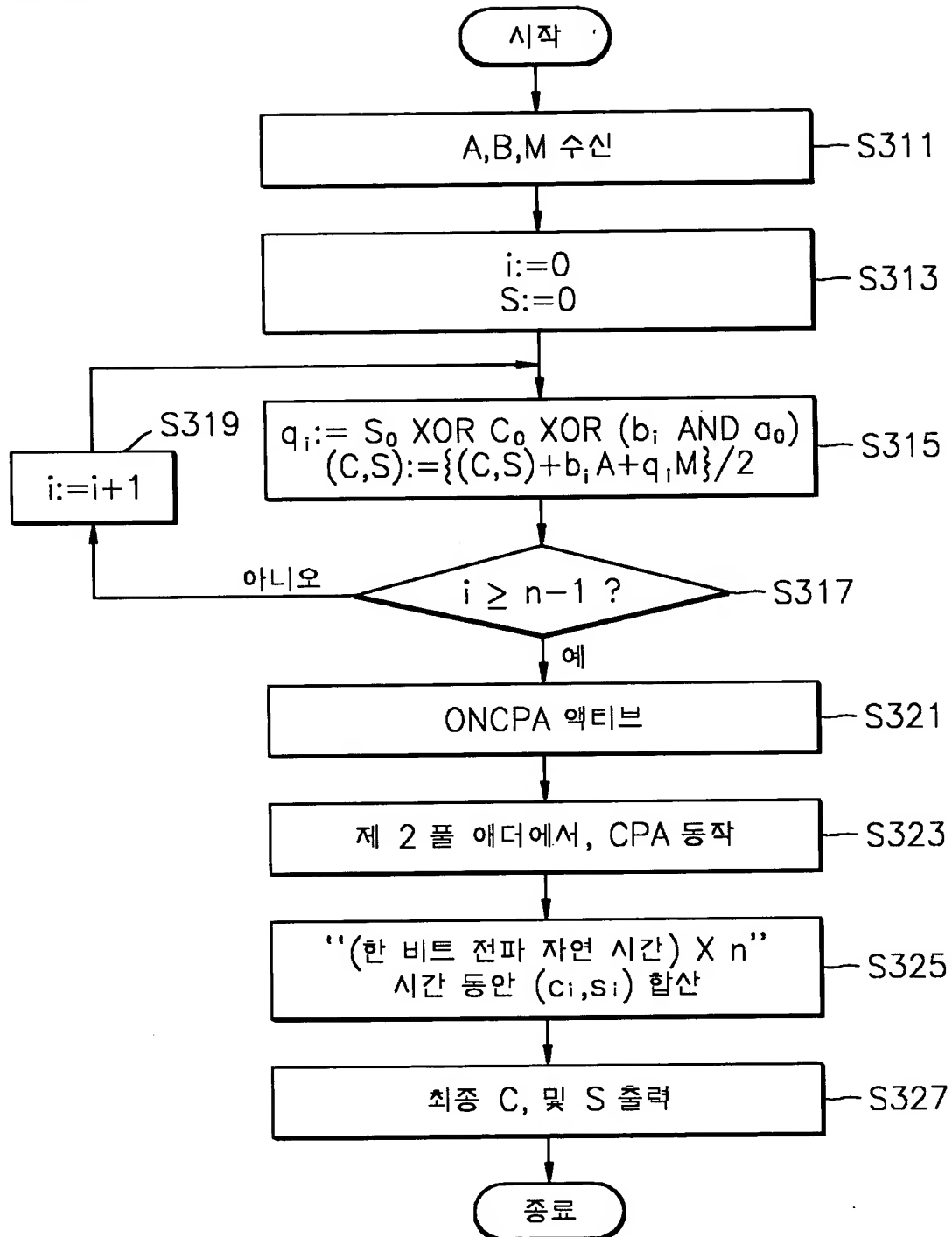




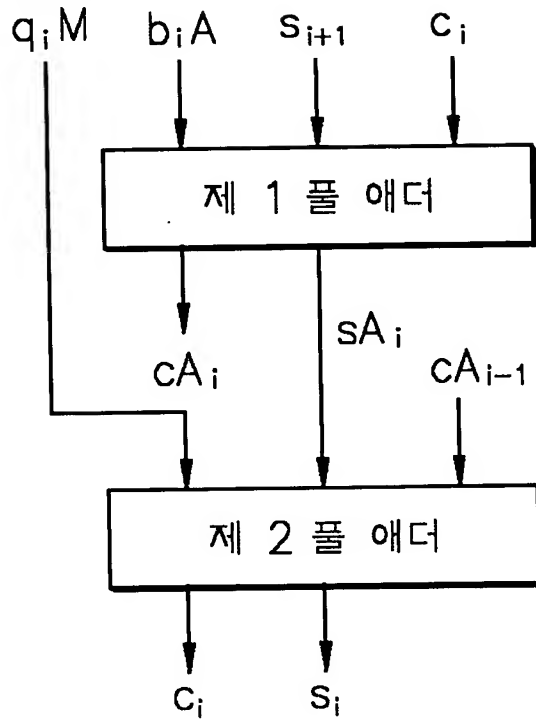
【도 2】



【도 3】



【도 4】



【도 5】

